

UBIKA

WAAP AS A CONTAINER

SCALE YOUR SECURITY ON KUBERNETES



UBIKA WAAP as a Container, is the new, ultra-scalable WAAP that protects your web applications and APIs from within your Kubernetes clusters.

The 3 Key Challenges of Transitioning to Distributed Application Security

1

Service Continuity: A Challenge Under Pressure

Service continuity is under pressure today due to the rise in DDoS attacks and increasing infrastructure costs, which require companies to be both more resilient and more efficient. They must now rely on architectures capable of absorbing traffic spikes while optimizing resources, which drives the adoption of elastic models. In this context, Kubernetes has emerged as a standard, enabling the automation of application deployment, scalability, and resilience, and has thus become an essential pillar of modern service availability.

2

Adapting Security to Microservices Already in Production

The operational reality for most organizations is that their K8s clusters are already in production, microservices architectures are deployed, and CI/CD pipelines are running. Security must therefore often adapt to the existing environment.

3

Integrate WAAP into existing DevSecOps workflows

Faced with rapidly evolving architectures and threats, organizations are integrating security directly into the DevSecOps cycle to reduce human error and strengthen governance. Security policies become versioned code in Git, subject to review, visible, and traceable, ensuring a "no drift" principle where the actual state remains aligned with the defined state. In Kubernetes, this approach is implemented through "policy as code" and automation via CI/CD pipelines.

UBIKA WAAP as a Container: The Answer to Modern Challenges

With **UBIKA WAAP as a Container**, managed via CRDs, application protection becomes "**Security & Workflow as Code**" by design. Security and operational processes are treated as code, designed from the outset to be versioned, automated, and integrated into pipelines, ensuring consistency, governance, and scalability.

1

A security policy versioned and governed by Git

UBIKA WAAP as a Container allows you to define security policies declaratively, typically in YAML, directly within the Git repositories of applications or the platform.

In practice:

- Security rules become versioned artifacts
- Every change is logged
- All policies are readable, auditable, and traceable

Benefits:

- Total transparency: No rules hidden in a console or device
- Easier auditing: Who changed what, when, and why
- Dev/Sec/Ops alignment: A single source of truth

2

Changes Subject to Review, Just Like Application Code

With **UBIKA WAAP as a Container**, any change to the security policy can be subject to a standard process:

- Pull request / merge request
- Peer review (Dev, Sec, Platform)
- Validation before deployment

In practice:

- Add an anti-bot rule
- Modify a rate limit
- Enable API protection

Benefits:

- Reduced risk of human error
- Consistency of rules over time
- Empowerment of teams

3

The ability to roll back immediately in the event of an incident

In a mission-critical production environment, a change to a security rule can have unexpected effects, such as blocking false positives, reduced availability, and business impact. With **UBIKA WAAP as a Container**, a simple Git revert allows you to return to a known functional state.

Benefits:

- Instant rollback
- Reduced MTTR (Mean Time To Repair)
- Secure deployment to production

4

A guarantee of "no drift" between the desired state and the actual state

In traditional systems, configurations drift. There are manual changes in production, undocumented emergency fixes, and inconsistencies between environments.

UBIKA WAAP as a Container is integrated into a Kubernetes/GitOps framework:

- Treats Git as the source of truth
- Continuously enforces the desired state
- Detects any discrepancies

Benefits:

- No silent drift
- Constant alignment between code and reality
- Reproducible security across environments

5

Compliance with Kubernetes design patterns

UBIKA WAAP as a Container is designed to run natively in Kubernetes, adhering to its standards:

- Deployment via Helm charts
- Integration with Ingress Controllers
- Capability to run as a sidecar or service
- Configuration via CRDs (Custom Resources)

6

Better scalability, aligned with workloads

Thanks to this cloud-native integration:

- The WAAP scales with the pods/services
- No single point of congestion
- Automatic adaptation to traffic spikes (e.g., L7 DDoS)

Benefits:

- Automatic horizontal scalability
- Consistent performance with the application
- Optimized costs (no overprovisioning)

7

Observability integrated into the existing ecosystem

UBIKA WAAP as a Container integrates seamlessly with standard Kubernetes tools:

- Prometheus for metric export
- Grafana for visualization
- Centralized pipeline (ELK, SIEM, etc.) for log management

Examples of metrics:

- Blocked requests
- Types of attacks
- Targeted endpoints
- Behavioral anomalies

Benefits:

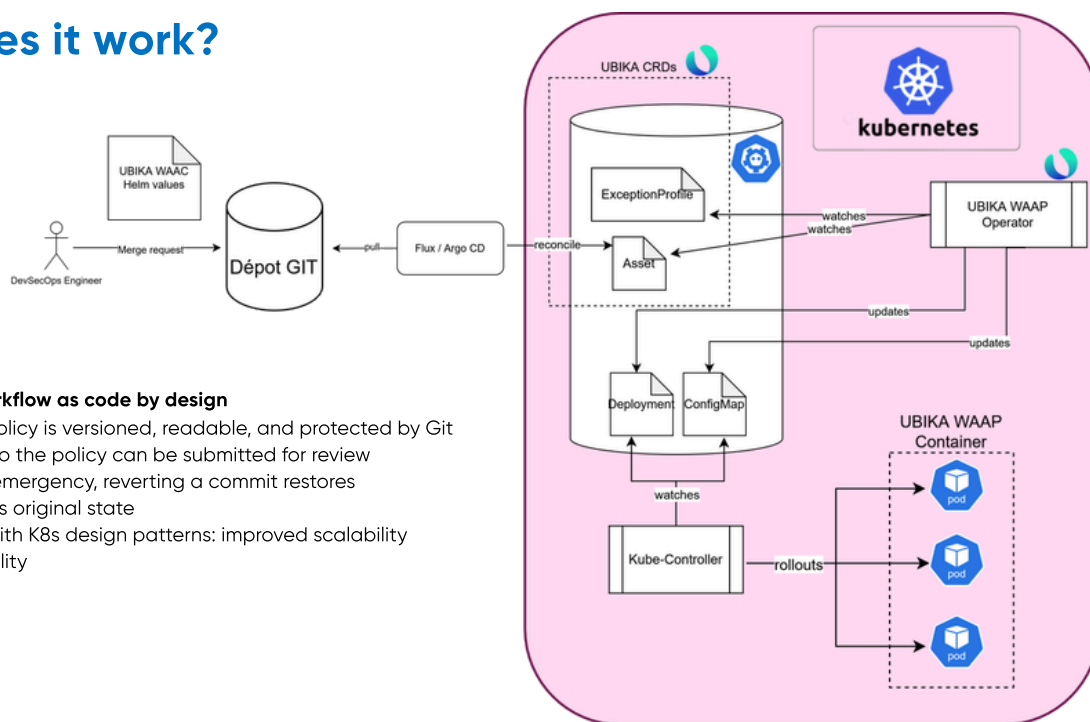
- Unified visibility into applications and security
- Rapid correlation in the event of an incident
- Integration without additional tooling layers

8

Protection for your web applications and APIs outside of microservices and Kubernetes

UBIKA WAAP as a Container also protects applications that are not microservices or are not running on Kubernetes. Upstream traffic cleaning ensures your apps do not consume excessive bandwidth. As a result, **UBIKA WAAP as a Container** enables your infrastructure to evolve smoothly and securely toward Kubernetes.

How does it work?



✓ Security & workflow as code by design

- ✓ The security policy is versioned, readable, and protected by Git
- ✓ Any changes to the policy can be submitted for review
- ✓ In case of an emergency, reverting a commit restores the policy to its original state
- ✓ Compliance with K8s design patterns: improved scalability and observability